IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors: Seisuke MORIOKA

Serial No. 09/423,415

Filed: November 5, 1999

STATEMENT OF ACCURACY

Title: IMAGE PROCESSOR AND IMAGE PROCESSING METHOD

Hon. Commissioner of Patents and Trademarks
Washington D.C. 20231

Sir:

The undersigned hereby states that the attached English language patent application is an accurate translation of the Japanese language application filed on May 7, 1997 and assigned Serial No. JP 9-116772.

Date: June 12, 2003 Signed: Tomoko Tokita

Printed Name: Tomoko TOKITA

(Translation)

[Document Name]	Patent Application
[Reference Number]	[redacted]
[Filing Date]	May 7, 1997
[Addressed to]	Director-General of the Patent Office
[International Patent Classification]	G06T 11/00 1/60 15/00
[Title of Invention]	IMAGE PROCESSING UNIT AND IMAGE PROCESSING METHOD
[Number of Claims]	9

[Inventor]

[Address]	c/o Sega Enterprises, Ltd. 2-12, 1-chome, Haneda, Ohta-ku, Tokyo
[Name]	Seisuke Morioka

[Applicant]

[ID Number]	000132471
[Name]	Sega Enterprises, Ltd.
[Address]	2-12, 1-chome, Haneda, Ohta-ku, Tokyo

[Agent]

[ID Number]	100079108
[Patent Attorneys]	
[Name]	Yoshiyuki Inaba

[Assigned Agent]

[ID Number]	100080953

[Patent Attorney]

[Name] Katsuro Tanaka

[Assigned Agent]

[ID Number] 100093861

[Patent Attorney]

[Name] Shinji Ohga

[Claim of Priority Based on Prior Application]

[Application Number]

[Date of Application]

[Document Name] Specification

[Title of the Invention] IMAGE PROCESSING UNIT AND IMAGE PROCESSING METHOD

[Claims]

[Claim 1]

An image processing unit comprising a first storage device for storing texture data, a second storage device for storing a part of said texture data, and a processing section for executing image processing based on the texture data in said second storage device, where said processing section updates the texture data by reading the texture data from said first storage device and writing that data to said second storage device in a predetermined case.

[Claim 2]

The image processing unit according to Claim 1, characterized in that said first storage device stores data including compressed texture data, and said processing section further comprises a data decompression circuit for decompressing the read texture data so as to write the decompressed data to said second storage device.

[Claim 3]

The image processing unit according to Claim 2, characterized in that said processing section further comprises a first-in-first-out storage device for receiving the read texture data, temporarily storing this data, and outputting the data to said data decompression unit.

[Claim 4]

The image processing unit according to Claim 1, characterized in that said processing section further comprises a palette transformation circuit for executing palette transformation when the texture data is updated.

[Claim 5]

The image processing unit according to Claim 1, characterized in that said processing section further comprises a mip map generation circuit for generating a mip map when the texture data is updated.

[Claim 6]

An image processing method using a first storage device for storing texture data and a second storage device for storing a part of said texture data so as to execute image processing based on said texture data in said second storage device, comprising an updating step for updating the texture data by reading the texture data from said first storage device and writing that data to said second storage device in a predetermined case.

[Claim 7]

The image processing method according to Claim 6, further comprising a data decompression step for decompressing the read texture data when the data stored in said first storage device is compressed texture data, characterized in that the decompressed data is written to said second storage device.

[Claim 8]

The image processing method according to Claim 6, further comprising a palette transformation step for executing palette transformation when the texture data is updated.

[Claim 9]

The image processing method according to Claim 6, further comprising a mip map generation step for generating a mip map when the texture data is updated.

[Detailed Description of the Invention]

[0001]

[Field of the Invention]

The present invention relates to an image processing unit and image processing method for computer graphics.

[0002]

[Related Art]

Providing real and gorgeous images in a computer graphic (CG) system using texture mapping is demanded. The easiest means for generating such real and gorgeous images is using a large volume of texture data.

[0003]

[Problems to be Solved by the Invention]

However, in a system which executes high quality texture mapping such as tri-linear mapping, very high-speed access is demanded for the texture buffer, and to

use a large volume of data, a large capacity high-speed storage device must be provided. This requires enormously high cost, which makes it difficult to provide a large capacity storage device.

[0004]

With the foregoing in view, it is an object of the present invention to provide an image processing unit and image processing method which can generate superb images using a relatively small capacity texture buffer.

[0005]

[Means for Solving the Problems]

An image processing unit according to the present invention comprises a first storage device for storing texture data, a second storage device for storing a part of the above texture data, and a processing section for executing image processing based on the texture data in the above second storage device, where the above processing section updates the texture data by reading the texture data from the first storage device and writing that data to the second storage device in a predetermined case.

[0006]

Here the predetermined case is a case when updating the texture data is necessary, such as the case when texture data not stored in the second storage device must be used. The texture data may be updated in advance when processing capability is sufficient. The storage devices are not only semiconductor memories but include such external devices as HDD and CD-ROM.

[0007]

The image processing unit according to the present invention is characterized in that the above mentioned first storage device stores data including compressed texture data, and the above mentioned processing section further comprises a data decompression circuit for decompressing the read texture data so as to write the decompressed data to the second storage device.

[0008]

Here the texture data to be stored in the first storage device can be non-compressed data, data containing both compressed and non-compressed data, or compressed data.

[0009]

The image processing unit according to the present invention is characterized in that the above mentioned processing section further comprises a first-in-first-out storage device for receiving the read texture data, temporarily storing this data, and outputting the data to the above mentioned data decompression circuit.

[0010]

The image processing unit according to the present invention is characterized in that the above mentioned processing section further comprises a palette transformation circuit for executing palette transformation when the texture data is updated.

[0011]

The image processing unit according to the present invention is characterized in that the above mentioned processing section further comprises a mip map generation circuit for generating a mip map when the texture data is updated.

[0012]

An image processing method according to the present invention uses a first storage device for storing texture data and a second storage device for storing a part of the texture data, so as to execute image processing based on the texture data in the second storage device, and comprises an updating step for updating the texture data by reading the texture data from the first storage device and writing that data to the second storage device in a predetermined case.

[0013]

The image processing method according to the present invention further comprises a data decompression step for decompressing the read texture data when the data stored in the first storage device is compressed texture data, characterized in that the decompressed data is written to the second storage device.

[0014]

The image processing method according to the present invention further comprises a palette transformation step for executing palette transformation when the texture data is updated.

[0015]

The image processing method according to the present invention further comprises a mip map generation step for generating a mip map when the texture data is updated.

[0016]

[Mode for Carrying Out the Invention]

Embodiment 1

Equipment and method of an embodiment 1 of the present invention will now be explained. The embodiment 1 of the present invention has a mechanism to update texture in real-time at high-speed.

[0017]

Fig. 1 is a block diagram of an image processing unit in accordance with the embodiment 1 of the present invention. In Fig. 1, 1 is a CPU (Central Processing Unit) which manipulates objects in a virtual space, receives information thereof, and executes various controls. 2 is a geometry processor, which executes the coordinate transformation of polygons, such geometric transformation (vector arithmetic operations) as clipping and perspective transformation, and luminance calculation in three dimensional computer graphics at high-speed. 2a is a polygon material light buffer RAM, which is a buffer for storing effective polygon data, material data and light data for one frame when the geometry processor 2 executes processing. A polygon is a polygon constituting a three-dimensional body in a virtual space. A breakdown of data to be stored in the buffer memory 2a follows.

[0018]

Link Information, coordinate information and other attribute information of polygons.

LINK X, LINK Y, X, Y, iz, Tx, Ty, Nx, Ny, Sign Nz, Alpha, Light ID, Material ID etc.

[0019]

Material information

Depth enable, Depth function, Depth density, Texture enable, Fog enable, translucency enable, texture type, texture function, offset x, y, size x, y, repeat x, y, mirror x, y, color id, Sine, Material specular, Material emission, Polygon color, Texture mode, blend mode, etc.

[0020]

Light information

Light Position, Light Direction, Light Type, Attenuation, Cutoff, Spotexp. Light Color, Light Ambient, etc.

[0021]

3 is a fill processor for executing hidden surface removal processing. The fill processor 3 fills a polygon in an area, and determines each information on the polygon which is closest to the viewer for each pixel.

[0022]

4 is a texture processor. The texture processor 4 pastes texture on each pixel in an area. Texture mapping is a processing for creating an image by pasting

(mapping) patterns (texture), which are defined separately from the shape, on the surface of an object for which shape has been defined. 4a is a texture RAM, where a texture map for the texture processor 4 to execute processing is stored.

[0023]

5 is a shading processor. Shading is a method to express the shadow of an object comprised of polygons while considering the normal line vector of a polygon, position and color of a light source, position of view point, direction of line of sight, and other factors. The shading processor 5 determines the luminance of each pixel in an area. 5a is a frame buffer for storing image data on one screen. Data is sequentially read from the frame buffer 5a, and after the digital data is converted to analog signals, the analog signals are supplied to such displays as a CRT, liquid crystal display and plasma display, which are not depicted here.

[0024]

6 is a program work polygon buffer RAM for storing the programs of the CPU1 and commands to the graphic processor (e.g. database of polygons, display lists). This buffer memory 6 is a work memory of the CPU1 as well.

[0025]

The fill processor 3, texture processor 4 and shading processor 5 execute rendering for creating pictures using models defined in the virtual space coordinates. In rendering, each area is processed sequentially from the upper left of the screen. Rendering processing is repeated for the number of areas.

[0026]

Now details on the image processing unit in accordance with the embodiment 1 of the present invention will be described with reference to the function block diagrams in Fig. 2 to Fig. 5.

[0027]

Fig. 2 is a functional block diagram of the geometry processor 2. In Fig. 1, 21 is a data dispatcher, which reads a command from the buffer RAM 6, analyzes the command, controls a vector engine 22 and clipping engine 24 based on the analysis result, and outputs the processed data to a sort engine 27.

[0028]

22 is the vector engine for executing vector arithmetic operations. Vectors to be handled are stored in a vector register 23.

[0029]

23 is the vector register for storing vector data for the vector engine 22 to operate.

[0030]

24 is the clipping engine for executing clipping.

[0031]

25 is a Y-sort INDEX, for storing the Y index for a sort engine 27 to execute Y sorting.

[0032]

26 is an X-sort INDEX, for storing the X index for the sort engine 27 to execute X sorting.

[0033]

27 is the sort engine (sort engine) for searching a polygon to enter the target fragment in the buffer 6 by executing X sorting and Y sorting. The searched polygon is stored in the buffer memory 2a, and is also sent to the fill processor 3 for rendering. The sort engine 27 also controls a polygon TAG 28 and a polygon cache 34.

[0034]

28 is the polygon TAG, which is a buffer for storing the TAG of the polygon cache 34.

[0035]

Fig. 3 is a functional block diagram of the geometry processor 2. In Fig. 3, 31 is a cache controller for controlling the later mentioned material caches 42, 45, 51b, 52a and 53a, and a light cache 51a.

[0036]

32 is a material TAG for storing the TAG of the later mentioned material caches 42, 45, 51b, 52a and 53a, and light cache 51a.

[0037]

33 is a light TAG, which is a buffer for storing the TAG of the later mentioned light cache 51a.

[0038]

34 is a polygon cache, which is a cache memory of polygon data.

[0039]

35 is an initial parameter calculator, for determining the initial value of DDA.

[0040]

36 is a Z comparator array for executing Z comparisons between polygons for hidden surface removal processing, and also for padding the polygon ID and internal ratios t0, t1 and t2. The Z comparator array 36 comprises 8x8=64 units of Z comparators. Since these Z comparators operate in parallel, 64 pixels can be simultaneously processed. Each Z comparator stores data on a polygon. For example, a polygon ID, iz, t0, t1, t2, window, stencil and shadow are stored.

[0041]

37 is a vertex parameter buffer, which is a buffer for storing parameters of a vertex of a polygon. Corresponding to the Z comparator array 36, the vertex parameter buffer has a size capacity for 64 polygons.

[0042]

38 is an interpolator for calculating the parameters of a pixel by interpolation based on the calculation result of the Z comparator array 36, t0, t1, t2, iz and the content of the vertex parameter buffer 37.

[0043]

Fig. 4 is a functional block diagram of the texture processor 4. In Fig. 4, 41 is a density calculator for calculating the blending ratio for fog or depth cueing.

[0044]

42 is a material cache for storing data on depth information. For example, Depth enable, Depth function, Depth density, Depth end z, Texture enable, and Fog enable are stored.

[0045]

43 is a window register, which is a buffer for storing information on windows. For example, kz, cz, fog function, fog density and fog end z are stored. 44 is an address generator for calculating addresses on a texture map based on the texture coordinates Tx, Ty and LOD.

[0046]

45 is a material cache for storing data on material. For example, translucency enable, texture type, offset x, y, size x, y, repeat x, y, mirror x, y, and color id are stored.

[0047]

46 is a TLMMI calculator (TLMMI: Tri-Linear Mip Map Interpolation) for executing tri-linear mip map interpolation, which is three dimensional interpolation. Mip map is a technique for anti-aliasing when texture mapping is executed, that is, a technique for removing texture jaggies. Mip map is based on the following principle. Originally, the color (luminance) of an object face to be projected to one pixel must be a mean value of colors of the corresponding mapping area. Otherwise, jaggies become outstanding, which drops the quality of texture dramatically. However, if a mean value is determined for each pixel, the calculation load becomes too high, which takes time to process, and which requires a high-speed processor. Mip map solves this problem. In mip map, a plurality of mapping data having a width which is a multiple of 2 are prepared in advance so as to simplify the tabulation of colors (luminance) of the mapping area corresponding to one pixel. A size of the entire mapping area corresponding to one pixel is between any two data of these plurality of data having a width which is a multiple of 2. By comparing these two data, the color of the corresponding mapping area is determined. For example, when screen A (x 1) and screen B (x 1/2) exist, the pixels of screen A and the pixels of screen B corresponding to each pixel of screen C (x 1/1.5) are determined respectively. At this time, the color of the pixel of screen C is a color between the pixel in screen A and screen B.

[0048]

47 is a color converter for executing color conversion at 4 bit texel.

[0049]

48 is a color palette where color information at 4 bit texel is stored. The color pallet 48 stores colors to be used for drawing graphics. Colors that can be used for one pixel are determined corresponding to the content of the color pallet 48.

[0050]

Fig. 5 is a functional block diagram of the shading processor 5. In Fig. 5, 51 is an intensity processor for calculating the luminance for polygons after texture mapping.

[0051]

51a is a light cache for storing light information. For example, Light Position, Light Direction, Light Type, Attenuation, Cutoff, Spotexp. Light Color and Light Ambient are stored.

[0052]

51b is a material cache for storing information on material. For example, Sine, Material specular, material emission are stored.

[0053]

51c is a window register for storing information on windows. For example, Screen center, Focus, Scene ambient are stored.

[0054]

52 is a modulate processor for executing associations of polygon color and texture color, intensity modulation and fog processing.

[0055]

52a is a material cache for storing information on materials. For example, Polygon color, and Texture mode are stored.

[0056]

52b is a window register, which is a buffer for storing information on windows. For example, Fog Color is stored.

[0057]

53 is a blend processor for blending data on the color buffer 54 and writing the blended data to the color buffer 54. The blend processor 53 blends the current pixel color and the pixel color of the frame buffer based on the values of the blend rate register, and writes the blended data to the frame buffer of the bank indicated by the light bank register. With the blend processor 53, residual image processing can be executed.

[0058]

53a is a material cache for storing information on materials. For example, blend mode is stored.

[0059]

54 is a color buffer, which is an 8x8 size color buffer. The color buffer 54 has a double bank structure.

[0060]

55 is a plot processor for writing data on the color buffer 54 to the frame buffer 5a.

[0061]

56 is a bitmap processor for executing bit map processing.

[0062]

57 is a display controller, which reads data of the frame buffer 5a, supplies the data to DAC (Digital to Analog Converter), and displays the data on a display which is not depicted here.

[0063]

Fig. 6 is a block diagram of the equipment shown in Fig. 1 - Fig. 5, created for explanatory purposes. The image processing unit 101 is connected with the CPU1 and exchanges commands and data with the CPU1, and accesses an external storage device 102, such as a hard disk, a buffer memory 103, a texture buffer 104 and a frame buffer 5a. The texture buffer 104 stores texture maps. This texture buffer 104 executes high quality texture mapping in real-time, therefore a high-speed operation is required. The buffer memory 103 also stores compressed texture data along with other data. The buffer memory 103 is a general buffer memory, which has a slower speed and larger capacity then the texture buffer 104. The image processing unit 4 of the embodiment 1 of the present invention has a function for reading compressed texture data from the buffer memory 3, executing data

decompression, and writing that data to the texture buffer 104, in addition to ordinary processing.

[0064]

Fig. 7 is a block diagram of the image processing unit 101. Fig. 7 corresponds to the flow of processing. Numerals 112 - 119 are circuits for processing a conventional three dimensional computer graphic system. These circuits are cascaded. Numerals 111, 120 - 124, on the other hand, are circuits for executing the processing of the embodiment 1 of the present invention. These circuits are cascaded, and the final output is input to the texture generation circuit 117 via the texture RAM interface 124.

[0065]

A coordinate transformation circuit 112 receives a command from a later mentioned command analysis circuit 111 and executes coordinate transformation for generating images. A clipping circuit 113 clips a portion outside the field of view after the data is transformed to an eye coordinate system. A perspective transformation circuit 114 transforms the eye coordinate system to the screen coordinate system. A fill circuit 115 fills transformed images. A Z comparison circuit 116 compares Z values which indicate perspective, and executes hidden surface removal by the Z sort method. A texture generation circuit 117 generates texture based on the texture data. A color modulation circuit 118 and a blend circuit 119 adjust the color of images.

[0066]

A command analysis circuit 111 analyzes commands from the CPU1, distinguishes a command (data) for image generation and a command (data) for updating texture data, and channels the commands to the coordinate transformation circuit 112 and FIFO (First-In-First-Out) 120. The FIFO 120 is a first-in-first-out memory for absorbing the difference between the data read speed and the data decompression/write speed. FIFO is a memory used for data exchange between two parts which have different timing, and can adjust the difference even if timing is different at both ends of FIFO. A data decompression circuit 121 is a circuit for decompressing the compressed data which was input. A palette transformation circuit 122 is a circuit for executing palette transformation. A MIP MAP generation circuit 123 is a circuit for automatically generating a mip map. Mip map is a technique for anti-aliasing when texture mapping is executed, that is, a technique for removing jaggies. A texture RAM interface (I/F) 124 is an interface with a texture buffer (RAM) 104, which has a mechanism for mediating between ordinary image processing and updating texture.

[0067]

The operation will now be explained.

[0068]

Data compression is very effective for such image data as a texture map. For example, a full color (16,000,000 colors) image can be expressed by 8 bit per texel, since in most cases several hundred colors are actually used. In mip map, a plurality of mapping data having a width which is a multiple of 2 is prepared to simplify the tabulation of colors (luminance) of the mapping areas corresponding to

one pixel, and a mip map, which is a simple reduction of the original image, can be easily generated only if the original image is available. The image data often has the same color as the texel color of a nearby area where many repeat patterns exist, therefore a general compression algorithm, such as run length and slide dictionary, can easily be applied.

[0069]

If the data is stored in a compressed state, however, decompression processing is required to use the data, which takes time. Since texture data is accessed at high-speed, compressed data is not easy to use, and the data must be stored in a non-compressed state.

[0070]

With the foregoing in view, the embodiment 1 of the present invention makes high-speed access possible and improves the efficiency of memory use by storing data in a non-compressed state in the texture buffer 104, which must be accessed at high-speed, storing compressed data in the other buffer (buffer memory 103), and has a mechanism to update the texture buffer 104 in real-time.

[0071]

In Fig. 7, when a command (data) for image generation is transferred from the command analysis circuit 111 to the coordinate transformation circuit 112, texture generation, color modulation, and blending processing are executed based on coordinate transformation, clipping, perspective transformation, filling, Z comparison, and non-compressed data in the texture buffer 104.

[0072]

When a command (data) for updating the texture data is transferred from the command analysis circuit 111 to the FIFO 120, the following processing is executed.

[0073]

The FIFO 120 is a buffer to adjust the data read speed and data decompression/write speed. Data flow is different between reading and decompressing + writing. Since data read involves simple memory access, the data transfer speed is constant, such as 1 byte at 1 clock. But it is difficult to estimate how many bytes the data will become when the data is decompressed, because this depends on the compression method. For example, 1 byte data, which was input, may become 2 bytes or 0.5 bytes. As Fig. 8 shows, when 4 bytes of input data A, B, C and D become 8 bytes, A'0, A'1, A'2, A'3, BC', D'0, D'1 and D'2, if the bus transfer capability of output is twice that of input, it seems that data volume will be balanced and no data wait time occurs. However, in actual processing, wait time occurs since data flow is not constant, therefore extra wait states must be created. In the example in Fig. 8, for example, a wait state is required before reading the data B after reading the data A, and after reading the data D, respectively. Also two wait states are required after A'2 and A'3, which is data after decompression. So the FIFO 120 adjusts the timing. In the FIFO, data written at any time by the input side can be output at any time in the sequence of writing by the output side. Therefore the CPU1 can issue a command regardless the data decompression state in the image processing unit 101. The CPU1 does not have to distinguish a command (data) for image generation from a command (data) for updating the texture data.

[0074]

When the data decompression circuit 121 receives a command (compressed data) for updating the texture data, decompresses the data, and returns the data to a non-compressed state. The compressed data has been stored in the external storage device 102 or the buffer memory 103.

[0075]

In a CG system using texture mapping, a large volume of texture data must be used to generate real and gorgeous images most easily. Therefore there is an enormous number of types of compressed data pre-stored. On the other hand, there is much less texture data, which is actually used simultaneously. When the texture data in the texture buffer 104 is insufficient, the CPU1 issues a command to read the necessary compressed data from the buffer memory 103, decompress the data, and writes the data to the texture buffer 104. Since the CPU1 knows which texture data to use and what kind of texture data has been stored in the texture buffer 104, the CPU1 can easily issue the command by comparing the texture data to be used and the texture data stored in the texture buffer 104.

[0076]

Compression and decompressing processings are executed according to a commonly known algorithm. For example, run length encoding, sliding dictionary, Huffman, and discrete cosine transformation are used. For the format, JPEG combining discrete cosine transformation and Huffman, for example, is used.

[0077]

Run length encoding is an encoding method used when a repeat of the same pattern appears frequently. The length of a pattern is called "the run". When the run is long, the data length can be shortened by encoding the run before transferring, rather than transferring the pattern as is.

[0078]

Sliding dictionary is a method where previous data is stored, and this data is used for data compression. This method is effective for mesh patterns.

[0079]

Huffman is a code to be created such that the average number of bits per sample becomes a minimum when quantized sample sets are encoded. If Huffman is used, palette transformation can be executed with great flexibility. For example, when color is simple, such as in the case of images for animation, long patterns often appear while short patterns rarely appear. In this way, appropriate encoding is possible according to the nature of the image.

[0080]

JPEG (Joint Photographic Experts Group) is a color still picture compression system established by a group promoting the standardization of a color still picture compression system. JPEG is widely used for multimedia applications which handle still pictures, such as on personal computers.

[0081]

For the texture data after data decompression processing is executed, palette transformation is executed by the palette transformation circuit 122. Then the MIPMAP generation circuit 123 generates a mip map corresponding to the decompressed data.

[0082]

The texture data after data decompression processing is executed is written to the texture buffer 104 by the texture RAM interface 124. When the texture buffer 104 does not have an open area, the texture data overwrites other data. For example, data not frequency used, old data, or data used a long time ago, can be selected to be overwritten.

[0083]

In this way, according to the embodiment 1 of the present invention, virtually any texture data can be used, even if the capacity of the texture buffer is small.

[0084]

Since the texture data to be stored in the low speed buffer memory is compressed, memory space can be conserved.

[0085]

Also because the data decompression mechanism is included in the system, non-compressed data flows only over the high-speed bus of the texture buffer 104, and compressed data flows over the low speed bus, which allows sufficient use of the transfer capability of the buses.

[0086]

[Advantages of the Invention]

The present invention comprises a first storage device for storing texture data and a second storage device for storing a part of the texture data, and has an updating step for reading texture data from the first storage device and writing that data to the second storage device in a predetermined case, therefore many texture data can be used while using the small capacity second storage device. As a result, both low cost and high-speed processing can be implemented.

[0087]

The present invention also has a first-in-first-out storage device, which receives the read texture data, temporarily stores this data, and outputs the data to the data decompression circuit. Therefore a timing adjustment in data exchange is unnecessary.

[Brief Description of the Drawings]

[Fig. 1]

Fig. 1 is a functional block diagram of an image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 2]

Fig. 2 is a functional block diagram of a geometry processor of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 3]

Fig. 3 is a functional block diagram of a geometry processor of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 4]

Fig. 4 is a functional block diagram of a texture processor of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 5]

Fig. 5 is a functional block diagram of a shading processor of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 6]

Fig. 6 is a functional block diagram of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 7]

Fig. 7 is an internal block diagram of the image processing unit in accordance with the embodiment 1 of the present invention.

[Fig. 8]

Fig. 8 is an explanatory diagram on the data read, decompressing and write timing in the image processing unit in accordance with the embodiment 1 of the present invention.

[Description of Reference Numerals]

1	CPU
2	geometry processor
2a	polygon material light buffer RAM
3	fill processor
4	texture processor
4a	texture RAM
5	shading processor
5a	frame buffer
6	program work polygon buffer RAM
21	data dispatcher
22	vector engine
23	vector register
24	clipping engine
25	Y-sort INDEX
26	X-sort INDEX
27	sort engine
28	polygon TAG
31	cache controller
32	material TAG
33	light TAG
34	polygon cache
35	initial parameter calculator
36	Z comparator array
37	vertex parameter buffer

38 interpolator

41 density calculator

42 material cache

43 window register

44 address generator

45 material cache

46 TLMMI calculator

47 color comparator

48 color palette

51 intensity processor

51a light cache

51b material cache

51c window register

52 modulate processor

52a material cache

52b window register

53 blend processor

53a material cache

54 color buffer

55 plot processor

56 bitmap processor

57 display controller

[Document Name] Abstract

[Abstract]

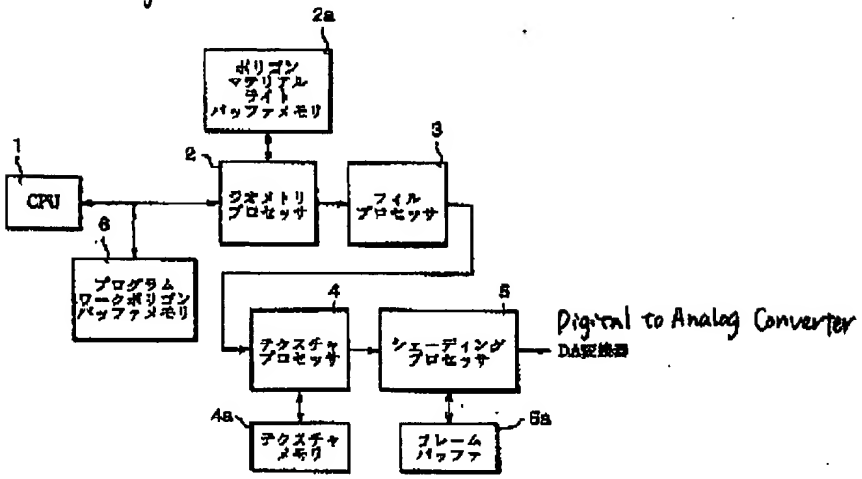
[Object]

To make it possible to use many polygons while using a relatively small capacity texture buffer, thereby generating superb images.

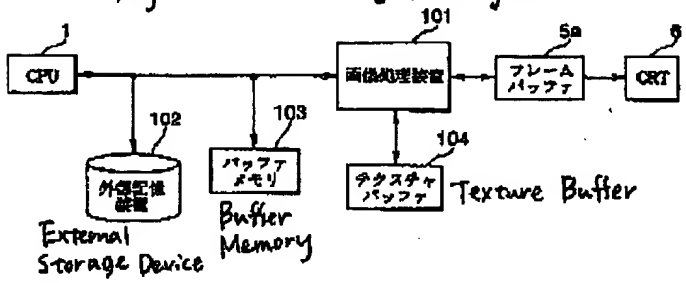
[Solving Means]

The present invention comprises a buffer memory 103 for storing compressed texture data and a high-speed texture buffer 104 for storing a part of texture data after executing decompression, so that when necessary texture data does not exist in the texture buffer 104, the texture data in the texture buffer 104 is updated by reading the texture data from the buffer memory 103, decompressing and writing the texture data to the texture buffer 104.

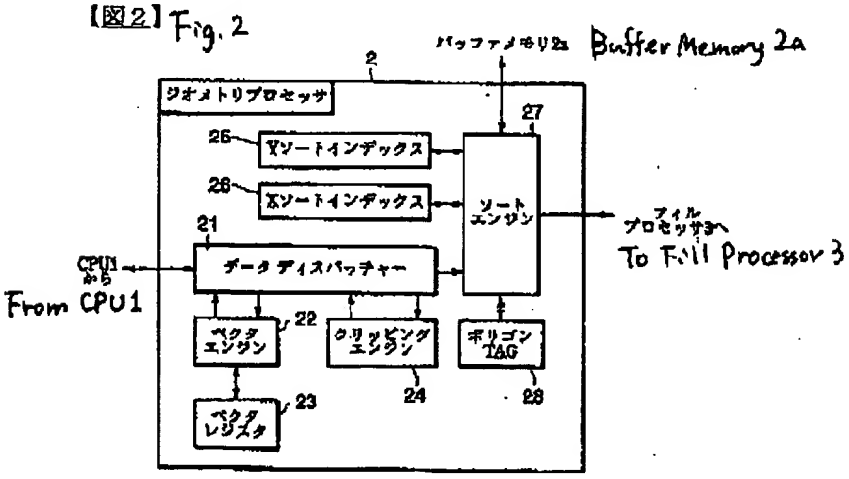
【図1】 Fig. 1



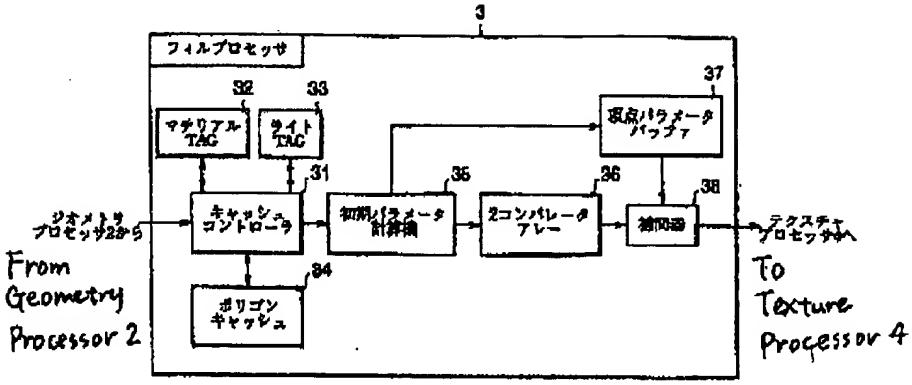
【図6】 Fig. 6



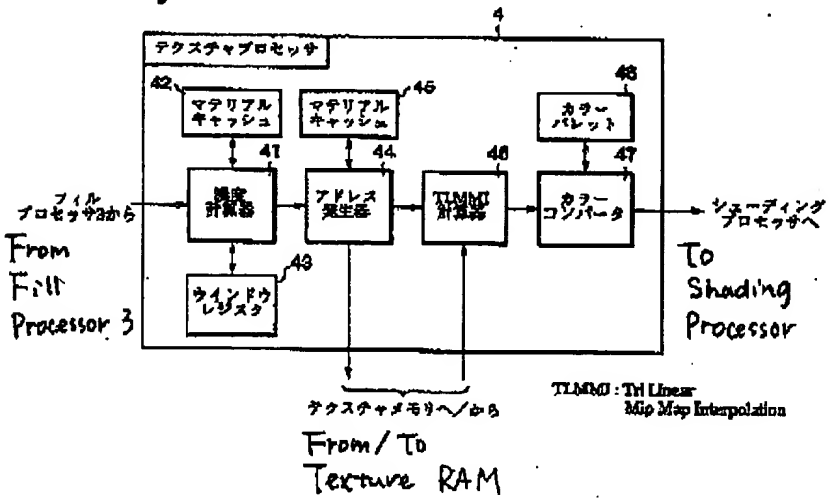
【図2】 Fig. 2



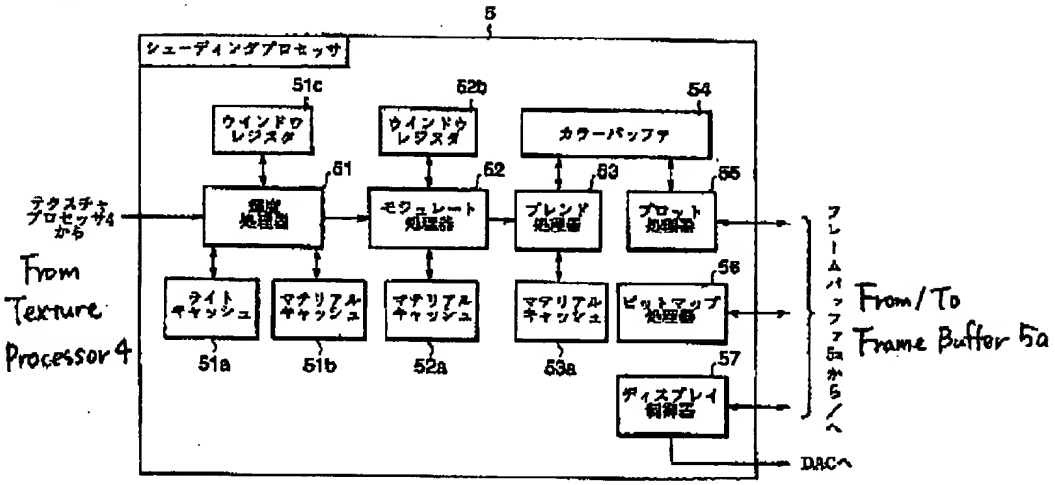
【図3】 Fig. 3



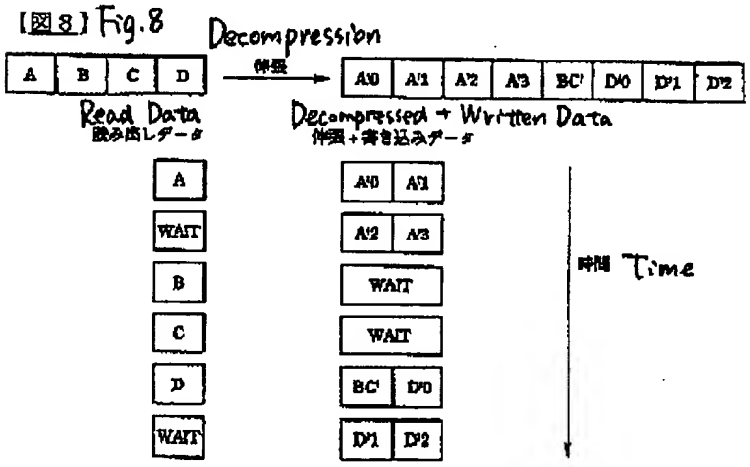
【図4】 Fig. 4



【図6】



【図8】Fig.8



【図7】 Fig. 7